# On Combining Ontologies and Rules

Matthias Knorr[1][0000−0003−1826−1498]

Universidade Nova de Lisboa, Portugal mkn@fct.unl.pt

**Abstract.** Ontology languages, based on Description Logics, and non-monotonic rule languages are two major formalisms for the representation of expressive knowledge and reasoning with it, that build on fundamentally different ideas and formal underpinnings. Within the Semantic Web initiative, driven by the World Wide Web Consortium, standardized languages for these formalisms have been developed that allow their usage in knowledge-intensive applications integrating increasing amounts of data on the Web. Often, such applications require the advantages of both these formalisms, but due to their inherent differences, the integration is a challenging task. In this course, we review the two formalisms and their characteristics and show different ways of achieving their integration. We also discuss an available tool based on one such integration with favorable properties, such as polynomial data complexity for query answering when standard inference is polynomial in the used ontology language.

## 1 Introduction

Though the central idea of Artificial Intelligence (AI) – creating autonomous agents that are able to think, act, and interact in a rational manner [69] – is far from being fullfilled, the results of the scientific advances of this major endavour already affect our daily lifes and our society. Such applications of AI technologies include, among others, clinical decision support systems in Healthcare, fraud dectection in financial transactions, (semi-)autonomous vehicles, crop and soil monitoring in agriculture, surveillance for border protection, but also in the form of personal assistants in cell phones. In these cases, technologies from different subareas of AI are applied, such as Learning, Vision, Reasoning, Planning, or Speech Recognition, but common to them is the requirement of efficiently processing and taking advantage of huge amounts of data, for example, for finding patterns in a data set, or for determining a conclusion based on the given data to aid making a decision.

Knowledge Representation and Reasoning (KRR) [14] in particular is an area in AI that aims at representing knowledge about the world or a domain of interest, commonly relying on logic-based formalisms, and that allows the usage of automated methods to reason with this knowledge and draw conclusions from it to aid in the beforementioned applications. Due to the structured formalization of knowledge, technologies based on KRR provide provenly correct conclusions and are commonly amenable to complement these with justifications or explanations for such derived results, which is important for accountabilility in applications where critical decisions need to be made, such as in Healthcare or in the financial sector. Yet, though KRR formalisms are responsible for many early success stories in the history of AI, namely in the form of expert systems, it has been pointed out that creating such formalizations often requires a major effort for each distinct such application [69].

As it turns out, a solution to the latter problem can be found in the area of the Semantic Web [39]. The central idea of the Semantic Web can be described as extending the World Wide Web, mainly targetted at human consumers, to one where data on the internet would be machine-processable, giving rise to advanced applications [12]. To achieve that goal, web pages are extended with machine-processable data written in standardized languages developed by the World Wide Web Consortium[1] (W3C) that allow the specification and identification of common terms in a uniformized way. In this context, the Linked Open Data initiative appeared leading to the publication of large interlinked datasets, utilizing these standardized languages, that are distributed over the Web. The resulting Linked Open Data Cloud[2] covers areas such as geography, government, life sciences, linguistics, media, scientific publications, and social networking, including data from, e.g., DBpedia [52], containing data extracted from Wikipedia, and prominent industry contributers such as the BBC, with accounted industry adopters, such as the New York Times Company or Facebook.

These developments facilitate the access and reuse of data and knowledge published on the Web in these standardized languages and thus also the creation of knowledge-intensive applications. The standards include the Resource Description Framework (RDF) [70] for describing information on directed, labelled and typed graphs, based on XML syntax; the Web Ontology Language (OWL) [40] for specifying shared conceptualizations and taxonomic knowledge; the Rule Interchange Format (RIF) [60] for expressing inferences structured in the form of rules, and SPARQL [38] for querying RDF knowledge. Among them, two are of particular interest for the representation of expressive knowledge, namely ontology languages and rule languages with different characteristics.

*Ontology languages* are widely used to represent and reason over formal conceptualizations of hierarchical, taxonomic knowledge. OWL in particular is founded on Description Logics (DLs) [5], which are commonly decidable fragments of first-order logic. Due to that, DLs are monotonic by nature, which means that acquiring new information does not invalidate previously drawn conclusions. Also, they apply the Open World Assumption (OWA), which means that no conclusions can be drawn merely based on the absence of some piece of information. They also allow us to reason over abstract relations between classes of objects, without the need to refer to concrete instances or objects, as well as to reason with unknown individuals, i.e., objects that are inferred to exist though they do not correspond to any known object in the represented knowledge. Dif-

[1] http://www.w3.org
[2] https://lod-cloud.net/

ferent DLs are defined based on varying combinations of logical operators that are admitted to be used, which allows one to choose a concrete language that balances between the required expressiveness of representation of the language and the resulting complexity of reasoning with it. This is why, in addition to the language standard OWL 2, which is very expressive but also comes with a high worst-complexity of reasoning, so-called profiles [61] of OWL 2 have been defined, each with different application areas in mind, that limit the admitted operators, but allow for polynomial reasoning.

*Rule languages* There is a large variety of rule languages, which in their essence can be described as expressing *IF - THEN* relationships. They are commonly divided into production rules, that can be viewed as describing conditional effects, and declarative rules that describe knowledge about the world. This is reflected in RIF in the sense that it is actually not a single language standard, but rather a number of different formats, so-called dialects, that can be integrated by means of RIF. Here, our interest resides on declarative languages to represent knowledge and reason with it, in that they represent inferences from premises to conclusions, commonly on the level of concrete instances. Among these languages, nonmonotonic rules as established originally in Logic Programming (LP) [10] are of particular interest. Such nonmonotonic rules employ the Closed World Assumption (CWA), i.e., if something cannot be derived to be true, it is assumed false. These conclusions may be revised in the presence of additional information, hence the term nonmonotonic. Nonmonotonic rules are thus well-suited for modelling default knowledge and exceptions, in the sense that commonly a certain conclusion can be drawn unless some exceptional characteristic prevents the conclusion, as well as integrity constraints that allow us to ensure that certain required specifications on the data are met.

*Integration* Due to the different characteristics of the two formalisms, their integration is often necessary in a variety of applications (see e.g., [65, 1, 74, 55, 45]). To illustrate these requirements, we present some examples of such use cases, starting with an example which we will revisit throughout the paper.

*Example 1.* Consider a customs service that needs to assess incoming cargo in a port for a variety of risk factors including terrorism, narcotics, food and consumer safety, pest infestation, tariff violations, and intellectual property rights. The assessment of the mentioned risks requires taking into consideration extensive knowledge about commodities, business entities, trade patterns, government policies and trade agreements. While some of this knowledge is external to the considered customs agency, such as the classification of commodities according to the international Harmonized Tariff System (HTS), or knowledge about international trade agreements, other parts are internal, such as policies that help determine which cargo to inspect (under which circumstances), as well as, for example, records on the history of prior inspections that allow one to identify importers with a history of good compliance with the regulations and those that require closer monitoring due to previous infractions.

In this setting, ontologies are a natural choice to model among others the international taxonomy of commodities, whereas rules are more suitable to model, e.g., the policies that determine which cargo to inspect. Notably, the distinct characteristics of the two formalisms are important to represent different parts of the desired knowledge. On the one hand, when reasoning about whether a certain importer requires close monitoring, the fact that we cannot find any previous infractions suffices to conclude by default that this importer is not a suspect. This aligns well with the CWA usually employed in nonmonotonic rules. On the other hand, the fact that we do not know whether a certain product is contained in a cargo container, should not allow us to conclude that it is not there. Here, the OWA of ontology languages is more suitable. At the same time, with ontology languages we are able to reason abstractly with commodities, for example listed in the manifest of a container without having to refer to concrete objects.

Another example can be found in clinical health care, where large ontologies, such as SNOMED CT,[3] are used for electronic health record systems and clinical decision support, but where nonmonotonic rules are required to express conditions such as dextrocardia, i.e., that the heart is exceptionally on the right side of the body. Also, when matching patient records with clinical trials criteria [65], modeling pharmacy data of patients using the CWA is cleary preferable (to avoid listing all the medications not taken), because usually it can be assumed that a patient is not under a specific medication unless explicitly known, whereas conclusions on medical conditions (or the absence of them) require explicit proof, e.g., based on test results. A further use-case can be found in the maintainance of a telecommunications inventory [45], where an ontology is used to represent the hierarchy of existing equipment (with a uniform terminology used for such equipment in different countries), and nonmonotonic rules are applied to detect different failures based on current sensor data.

However, the combination of ontologies and nonmonotonic rules is difficult due to the inherent differences between the underlying base formalisms and the way how decidability of reasoning is achieved in them (see, e.g., [27, 62, 28]). This raises several questions, such as:

- How to achieve such an integration given the inherent technical differences?
- What is a suitable topology, i.e., should both formalisms be on equal terms or should one's inferences only serve as input for the other (and not vice-versa)?
- How and when should Open or Closed World Assumption be applied?
- How to ensure decidability for the integration?
- Can we obtain efficient reasoning procedures?

In this lecture, we discuss how to answer these questions. For this purpose, after giving an overview on the two base formalisms, we review major approaches that have been introduced in the literature to tackle this problem, and we identify essential criteria along which these (and many other such) integrations have been

[3] http://www.ihtsdo.org/snomed-ct/

introduced. These criteria may also help choose the most suitable appproach among them for a concrete application. We then also discuss an available tool based on one such integration with favorable properties along the established criteria, including efficient reasoning procedures.

The remainder of this document is structured as follows. We provide necessary notions on Description Logic ontologies and nonmonotonic rules in Sections 2 and 3, respectively. We then discuss the principles along which integrations between the two formalisms have been developed in Section 4, and show, in Section 5 four such integrations in more detail. We proceed with presenting a tool for answering queries over such an integration in Section 6 and finish with concluding remarks in Section 7.

## 2 Description Logic Ontologies

Description Logics (DLs) are fragments of first-order logics for which reasoning is usually decidable. They are commonly used as expressive ontology languages, in particular as the formal underpinning for the Web Ontology Language (OWL) [40]. In this section, we provide an overview on DLs to facilitate the understanding of the remaining material and refer for more details and additional background to text books in related work [5, 41].

On a general level, DLs allow us represent information about objects, called individuals, classes of objects, called concepts, that share common characteristics, and relations between objects as well as between classes of objects, called roles. For example, $Bob$ is an individual, $Person$ a concept, and $hasSSN$ a role abbreviating "has social security number" allowing us to state that $Bob$ is a person and that he has a specific social security number.

More formally, DLs are defined over disjoint countably infinite sets of *concept names* $\mathsf{N_C}$, corresponding to classes of individuals, *role names* $\mathsf{N_R}$, that express relationships, and *individual names* $\mathsf{N_I}$, corresponding to objects, which, in terms of first-order logic, match unary and binary predicates, and constants, respectively.

Complex concepts (and complex roles) can be defined based on these sets and logical constructors (indicated here in DL syntax), which include the standard Boolean operators, i.e., negation ($\neg$), conjunction ($\sqcap$), and disjunction ($\sqcup$), universal and existential quantifiers ($\forall$ and $\exists$), as well as numeric restrictions on (binary) roles ($\leq mR$ and $\geq nR$ for numbers $m, n$ and role $R$), and inverses of roles ($R^-$ for role $R$).

For example, we can define concepts that represent persons that have a social security number ($Person \sqcap \exists hasSNN.\top$), or those whose heart is on the left- or on the right-hand-side of the body ($HeartLeft \sqcup HeartRight$), or those having at most one social security number ($\leq 1HasSNN.\top$) or the role "is the social security number of" as the inverse of the role of having a social security number ($hasSSN^-$).

It should be noted that each DL, i.e., each fragment of first-order logic, can be distinguished by the admitted constructors, by means of which complex concepts

| Name | Syntax | Semantics |
|---|---|---|
| inverse role | $R^-$ | $\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (y,x) \in R^\mathcal{I}\}$ |
| universal role | $U$ | $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ |
| top | $\top$ | $\Delta^\mathcal{I}$ |
| bottom | $\bot$ | $\emptyset$ |
| negation | $\neg C$ | $\Delta^\mathcal{I} \setminus C^\mathcal{I}$ |
| conjunction | $C \sqcap D$ | $C^\mathcal{I} \cap D^\mathcal{I}$ |
| disjunction | $C \sqcup D$ | $C^\mathcal{I} \cup D^\mathcal{I}$ |
| nominals | $\{a\}$ | $\{a^\mathcal{I}\}$ |
| universal restriction | $\forall R.C$ | $\{x \in \Delta^\mathcal{I} \mid (x,y) \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$ |
| existential restriction | $\exists R.C$ | $\{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}, (x,y) \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$ |
| Self concept | $\exists R.\mathsf{Self}$ | $\{x \in \Delta^\mathcal{I} \mid (x,x) \in R^\mathcal{I}\}$ |
| qualified number | $\leqslant nR.C$ | $\{x \in \Delta^\mathcal{I} \mid \sharp\{(x,y) \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\} \leq n\}$ |
| restrictions | $\geqslant nR.C$ | $\{x \in \Delta^\mathcal{I} \mid \sharp\{(x,y) \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\} \geq n\}$ |

**Fig. 1.** Syntax and semantics of role and concept expressions in $\mathcal{SROIQ}$ for an interpretation $\mathcal{I}$ with domain $\Delta^\mathcal{I}$, where $C, D$ are concepts, $R$ a role, and $a \in \mathsf{N_I}$.

and roles are inductively formed. An overview on the admitted constructors in the DL underlying the OWL 2 language, i.e., the description logic $\mathcal{SROIQ}$ [42], can be found in Figure 1.

Based on complex concepts and roles, we can define an ontology containing axioms that specify a taxonomy/conceptualization of a domain of interest. Such axioms allow us to indicate that a concept is subsumed by another, i.e., corresponding to a subclass relation, or that two concepts are equivalent ($\equiv$) as a shortcut for subsumption in both directions, and likewise that some role is subsumed by (or equivalent to) another role, as well as assertions indicating that some individual is an instance of a class (or a pair of individuals an instance of a role), and even certain role characteristics, such as transitivity, reflexivity or symmetry of roles.

For example, the following set of axioms

$$Person \sqsubseteq HeartLeft \sqcup HeartRight \tag{1}$$
$$Person \sqsubseteq \exists hasSNN.\top \tag{2}$$
$$isSSNOf \equiv hasSSN^- \tag{3}$$
$$Person(Bob) \tag{4}$$

indicates that every person has the heart on the left or on the right-hand-side (1), every person has a social security number (2), the relation referring to the person corresponding to an SSN is the inverse of the relation indicating the SSN of a person (3), and that Bob is a person (4).

Formally, an *ontology* $\mathcal{O}$ is a set of axioms divided into three components, a TBox, an RBox, and an ABox. A *TBox* is a finite set of *concept inclusions* of the form $C \sqsubseteq D$ where $C$ and $D$ are both (complex) concepts. An *RBox* is a finite set of *role chains* of the form $R_1 \circ \ldots \circ R_n \sqsubseteq R$, including *role inclusions* with $n = 1$, where $R_i$ and $R$ are (complex) roles, and of *role assertions*, allowing us to express that a role is transitive ($\mathsf{Tra}$), reflexive ($\mathsf{Ref}$), irreflexive ($\mathsf{Irr}$), symmetric ($\mathsf{Sym}$) or asymmetric ($\mathsf{Asy}$), and that two roles are disjoint ($\mathsf{Dis}$). Finally, an *ABox* is a finite set of *assertions* of the form $C(a)$ or $R(a, b)$ for concepts $C$, roles $R$, and individuals $a$, $b$.

The semantics of DL ontologies, i.e., their meaning, is defined in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty set $\Delta^{\mathcal{I}}$, its domain, and a mapping $\cdot^{\mathcal{I}}$ that specifies how the basic language elements are to be interpreted. More concisely, such a mapping ensures that

- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $a \in \mathsf{N_I}$;
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each $A \in \mathsf{N_C}$; and
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $R \in \mathsf{N_R}$.

That is, every individual is interpreted as one domain element, every concept as a subset of the domain, and every role as a set of pairs of domain elements. This mapping can be extended to arbitrary role and concept expressions as indicated in Figure 1 for the constructors usable in $\mathcal{SROIQ}$.

The semantics of axioms $\alpha$ is based on a satisfaction relation w.r.t. an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$, which is presented in Figure 2. In this case, we also say that $\mathcal{I}$ is a model of $\alpha$, and likewise that $\mathcal{I}$ is a model of ontology $\mathcal{O}$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$. It should be noted that an alternative way to define the semantics of ontologies is obtained by a direct translation into first-order logic. For example, (2) can be translated into

$$\forall x.(Person(x) \rightarrow (\exists y.hasSNN(x, y)))$$

Note that the syntax of DLs does not explicitly mention the variables introduced in their first-order translation. They are left implicit, and, due to the restriction to unary and binary predicates (concepts and roles) and their particular structure, no ambiguity exists. We refer for the details of this translation to [5].

Based on the semantics in place, we can consider reasoning and what kinds of inferences can be drawn. For example, from

$$Person \sqsubseteq \exists has.SpinalColumn \tag{5}$$
$$\exists has.SpinalColumn \sqsubseteq Vertebrate \tag{6}$$

we can conclude that $Person \sqsubseteq Vertebrate$ holds and we can draw this conclusion without needing to know any particular individual person. In addition, together with (4), we can also conclude that Bob in particular is a vertebrate, and, by (2) and (4), that there is some object which is a social security number, although we do not know precisely which one.

| Axiom $\alpha$ | Condition for $\mathcal{I} \models \alpha$ |
|---|---|
| $R_1 \circ \ldots \circ R_n \sqsubseteq R$ | $R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}}$ |
| $\mathsf{Tra(R)}$ | if $R^{\mathcal{I}} \circ R^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ |
| $\mathsf{Ref(R)}$ | $(x, x) \in R^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$ |
| $\mathsf{Irr(R)}$ | $(x, x) \notin R^{\mathcal{I}}$ for all $x \in \Delta^{\mathcal{I}}$ |
| $\mathsf{Dis(R,S)}$ | if $(x, y) \in R^{\mathcal{I}}$, then $(x, y) \notin S^{\mathcal{I}}$ for all $x, y \Delta^{\mathcal{I}}$ |
| $\mathsf{Sym(R)}$ | if $(x, y) \in R^{\mathcal{I}}$, then $(y, x) \in R^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$ |
| $\mathsf{Asy(R)}$ | if $(x, y) \in R^{\mathcal{I}}$, then $(y, x) \notin R^{\mathcal{I}}$ for all $x, y \in \Delta^{\mathcal{I}}$ |
| $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| $C(a)$ | $a^{\mathcal{I}} \in C^{\mathcal{I}}$ |
| $R(a, b)$ | $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ |
| $\neg R(a, b)$ | $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$ |
| $a \not\approx b$ | $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ |

**Fig. 2.** Semantics of $\mathcal{SROIQ}$ axioms for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$, where $C, D$ are concepts, $R, S$ are roles, and $a, b \in \mathsf{N_I}$.

In more general terms, a number of standard inference problems are defined for DLs. For example, we can test if a DL ontology $\mathcal{O}$ is *consistent* by determining if it has a model. We can also determine if a concept is *satisfiable* by finding a model $\mathcal{I}$ in which the interpretation of $C$ is not empty ($C^{\mathcal{I}} \neq \emptyset$). Also, concept $C$ *is subsumed by* concept $D$ w.r.t. $\mathcal{O}$, i.e., $C$ is a subclass of $D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{O}$. Moreover, we can check whether an individual $a$ is an *instance of a concept* $C$ w.r.t. $\mathcal{O}$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{O}$. Frequently, these reasoning tasks can be reduced to each other, e.g., $C$ is subsumed by $D$ iff $C \sqcap \neg D$ is unsatisfiable, which means that it suffices to consider only one of them. Also, based on these standard inference problems, advanced reasoning tasks are considered, such as *classification* which requires the computation of the subsumptions between all concept names in the ontology, or *instance retrieval* which determines all individuals that are true for a given concept.

Decidability is achieved by carefully restricting the admitted constructors such that models satisfy certain desirable properties. This is necessary, as the domain of an interpretation $\Delta^{\mathcal{I}}$ may be infinite. Note that DLs only allow the usage of unary and binary predicates which together with the admitted constructors ensures that basic DLs satisfy the tree-model property, i.e., domain elements in a model are related in a tree-shaped manner, which avoids cycles, and thus ensures decidability. For more expressive DLs such as $\mathcal{SROIQ}$ this does not suffice. In fact, if we were allowed to use the constructors in Figures 1 and 2 without additional restrictions, reasoning in $\mathcal{SROIQ}$ would be undecidable. Instead, role hierarchies, i.e., essentially axioms involving role inclusions, are restricted to so-called regular ones, which intuitively limits the usage of certain roles in, e.g., qualified number restrictions and role chains, so that a strict order can be established in between the roles. We refer for the details to [42].

While the applied restrictions ensure that $\mathcal{SROIQ}$ is decidable, this DL underlying the W3C standard OWL 2 is still very general and highly expressive, which in turn implies that reasoning with it is highly complex, in fact, it is N2ExpTime-complete [46]. But general purpose DL reasoners exist for this highly expressive language such as Fact++ [79], Pellet [71], Racer [37] or Konclude [75]. Still, one of the central themes in DLs is the balance between expressiveness and the computational complexity for the required reasoning tasks. This is why the profiles OWL 2 EL, OWL 2 QL and OWL 2 RL have been defined [61], i.e., considerable restrictions of the admitted language of OWL 2, for which reasoning is tractable. Here, we briefly overview these languages.

The DL underlying OWL 2 EL is $\mathcal{EL}^{++}$ [6]. Often, $\mathcal{EL}^{+}_{\bot}$, a large fragment of it is used that only allows conjunctions and existential restrictions of concepts, hierarchies of roles, and disjoint concepts. $\mathcal{EL}^{+}_{\bot}$ is tailored towards reasoning with large conceptual models, i.e., large TBoxes, and used in particular in large biomedical ontologies, and specifically tailored, highly efficient reasoners such as ELK [47] exist.

*DL-Lite*$_R$, one of the languages of the *DL-Lite* family [4] which underlies OWL 2 QL, admits in addition inverse roles and even disjoint roles, but, in exchange, it only allows for simple role hierarchies, no conjunction (on the left-hand side of axioms), and it imposes limitations on the usage of existential restrictions in particular on the left-hand side of inclusion axioms. The focus of this profile is on answering queries over large quantities of data, and combining relational database technology in the context of ontology-based data access [81].

Finally, OWL 2 RL, which builds on Description Logic Programs [36], is defined in a way that avoids inferring the existence of unknown individuals (e.g., a social security number though we do not know which) as well as nondeterministic knowledge, for the sake of efficient reasoning. OWL 2 RL allows more of the available DL constructors than the other two OWL profiles, but their usage is often restricted to one side of the inclusion axioms. This makes efficient RDF and rule reasoners applicable such as RDFox [64].

## 3 Nonmonotonic Rules

Nonmonotonic rules in Logic Programming (LP) have been intensively studied in the literature and a large body of theoretical results for different semantics of such rules has been presented (see e.g. [59]). Among them, the most widely used today are the answer set semantics [32] and the well-founded semantics [31], for which efficient implementations exist. In this section, we give a brief overview on nonmonotonic rules and these two semantics in particular, as they are essential for the integration of ontologies and rules.

In a broad sense, rules are used to represent that if a certain condition is true, then so is the indicated consequence. This includes the usage of default negation (in the condition) using operator **not**, which can be used to represent that something holds unless some condition is verified. We can for example state a rule that indicates that, for any vertebrate, the heart is on the left-hand-side of the body unless it is explicitly known to be on the right-hand-side:

$$HeartLeft(x) \leftarrow Vertebrate(x), \mathbf{not}\ HeartRight(x) \tag{7}$$

This rule is in principle applicable to arbitrary individuals due to the usage of variables, in this case $x$, and it intuitively allows us to infer for any $x$ that $HeartLeft(x)$ holds unless $HeartRight(x)$ holds.

In terms of syntax, we note that the basic building blocks of rules are rather easy to define. All we need is a set of predicates of some arity, i.e., the number of arguments they admit, and a set of terms over constants and variables as these arguments. These, in turn, allow us to define rules as implications over such atoms possibly with default negation in the condition.

More formally, we consider disjoint sets of constants $\Sigma_c$, variables $\Sigma_v$, and predicate symbols $\Sigma_p$.[4] Then a *term* is a constant $c \in \Sigma_c$ or a variable $v \in \Sigma_v$, and an *atom* is of the form $p(t_1, \ldots, t_n)$, where $p$ is an $n$-nary predicate symbol in $\Sigma_p$ and the $t_i$, for $i = 1, \ldots, n$ and $n \geq 0$, are terms. Atoms with $n = 0$ are also commonly called propositional. Such atoms allow us to express that certain n-nary relations hold, or, using default negation **not**, that they do not hold in absence of information to the contrary. Atoms and default negated atoms are also called *literals*.

Logic programs then consist of rules that combine literals in logic formulas of a specific form. More precisely, a *disjunctive (logic) program* $P$ consists of finitely many (implicitly universally quantified) *disjunctive rules* of the form

$$H_1 \vee \cdots \vee H_l \leftarrow A_1, \ldots, A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m \tag{8}$$

where $H_k$, $A_i$, and $B_j$ are atoms. Such a rule can be divided into the *head* $H_1 \vee \cdots \vee H_l$ and the *body* $A_1, \ldots, A_n, \mathbf{not}\ B_1, \cdots, \mathbf{not}\ B_m$, where "," represents conjunction. Hence such a rule is to be read as "If $A_1$ and ... and $A_n$ are true and $B_1$ and ... and $B_m$ are not, then (at least) one of $H_1$ to $H_l$ is".

We also identify the elements in the head and in the body of a rule with the sets $\mathcal{H} = \{H_1, \ldots, H_l\}$, $\mathcal{B}^+ = \{A_1, \ldots, A_n\}$, and $\mathcal{B}^- = \{B_1, \ldots, B_m\}$, where $\mathcal{B} = \mathcal{B}^+ \cup \mathbf{not}\ \mathcal{B}^-$, and we occasionally abbreviate rules with $\mathcal{H} \leftarrow \mathcal{B}^+, \mathbf{not}\ \mathcal{B}^-$ or even $\mathcal{H} \leftarrow \mathcal{B}$. Note that, in line with this set notation, the disjunction in the head and the conjunction in the body are commutative, i.e., the order of the elements in the head and in the body does not matter.

There are a number of different kinds of rules which yield a different expressiveness depending on how many and which literals are allowed in the head and the body of each rule, and we recall them in the following, as different integrations of ontologies and nonmonotonic rules admit different kinds of such rules. *Normal rules* do only admit one atom in the head, whereas *positive rules* do

[4] Please note that often also function symbols are introduced in the literature of LP, but since they jeopardize decidability of reasoning, and usually are not considered in integrations of ontologies and nonmonotonic rules, we do omit them here.

not admit default negation, and normal and positive programs can be defined accordingly. In particular, if the body of a rule is empty, i.e., $n = m = 0$, the rule is called a *fact*, and if, alternatively, the head is empty, i.e., $l = 0$, a *constraint*.

Constraints are an interesting modelling feature of nonmonotonic rules, as they allow to impose restrictions on the presence of certain information:

$$SSN\_OK(x) \leftarrow hasSSN(x, y) \tag{9}$$

$$\leftarrow Person(x), \mathbf{not}\ SSN\_OK(x) \tag{10}$$

This states that if $x$ has a (known) social security number $y$, then $x$'s social security number status is fine (9), and that there can be no person $x$ whose social security number status is not fine (10). Note that the DL axiom (2) is not an alternative as it does not impose a restriction, but rather allows us to infer that there is a social security number, though we do not know which.

Here, unlike for DLs, no restriction on the syntax of predicates is made, which means that reasoning with such programs would be undecidable when working with an infinite domain. Commonly, this is prevented by ensuring that variables in the rules can only be instantiated with “known values”, i.e., that are known in the program. Such rules are called *safe* which, formally, is the case if each variable in a rule of the form (8) occurs in an atom $A_i$ for some $i$, $1 \leq i \leq n$, and we assume in the following that all rules are safe. For example, the rules (7), (9), and (10) are safe, whereas the rule

$$\leftarrow Person(x), \mathbf{not}\ hasSSN(x, y)$$

is not due to $y$ not occurring in any positive literal in the rule body.

This restriction to known individuals seems arguably severe in comparison to DLs where we can reason over an infinite domain and unknown individuals. On the other hand, the following example rule taken from the context of Example 1

$$\begin{aligned} CompliantShpmt(x) \leftarrow\ & ShpmtCommod(x, y), HTSCode(y, z), \\ & ShpmtDeclHTSCode(x, z) \end{aligned}$$

states that $x$ is a compliant shipment, if $x$ contains $y$ whose harmonized tariff code is $z$ and $x$ is declared to contain $z$. This can be easily expressed as a rule, whereas a representation via DLs proves difficult due to the fact that, if viewed as a graph, the variable connections established in the rule provide one link from $x$ to $z$ via $y$ and one direct link, which is not tree-shaped. Hence, both formalisms indeed differ as to what can be represented.

We next proceed by giving an overview on the two standard semantics for such nonmonotonic rules, answer set semantics and well-founded semantics.

## 3.1 Answer Set Semantics

Answer Set Programming (ASP) is a declarative programming paradigm tailored towards the solution of large combinatorical search problems. The central idea is to encode the given problem in a declarative way using rules and use one of the efficient answer set solvers available, such as clasp [29] or DLV [3], to determine the answer sets corresponding to the solutions of the problem. The approach builds on the answer set semantics [32] which is a two-valued nonmonotonic declarative semantics for logic programs with close connections to other logic-based formalisms such as SAT, Default Logic, and Autoepistemic Logic.

While the full ASP language comes with a number of additional syntactic constructs beyond the syntax of programs we have presented so far (cf. the ASP-Core-2 Input Language Format [18]), we limit our considerations to normal programs for the sake of readability and since this suffices to convey the main ideas.

The models considered in this semantics, called answer sets, are represented by a set of atoms occurring in a given program, that are true. Those atoms from the program not occurring in the answer set are false. Since rules may contain variables, rules are grounded first, i.e., all variables are instantiated with constants occurring in the program in all possible ways, and the set of all ground instances of the rules of a program $P$ is denoted by $\mathsf{ground}(P)$.

*Example 2.* Consider program $P$ consisting of rule $hasSSN(Bob, 123) \leftarrow$ together with (9). Then $\mathsf{ground}(P)$ consists of:

$$\begin{aligned}
SSN\_OK(Bob) &\leftarrow hasSSN(Bob, 123)\\
SSN\_OK(Bob) &\leftarrow hasSSN(Bob, Bob)\\
SSN\_OK(123) &\leftarrow hasSSN(123, Bob)\\
SSN\_OK(123) &\leftarrow hasSSN(123, 123)\\
hasSSN(Bob, 123) &\leftarrow
\end{aligned}$$

Of course, state-of-the-art ASP solvers will only keep the first of the grounded rules in such a situation, as the body atom in the other rules can never be true anyway. Either way, it is clear that $\{SSN\_OK(Bob), hasSSN(Bob, 123)\}$ is a model if we treat these rules as implications in first-order logic.

Now, the main idea of answer sets builds on guessing a model and checking that it satisfies a certain minimality criterion, namely, that there is some rule that supports the truth of some atom in an answer set (which is not the case, e.g., for $hasSSN(Bob, Bob)$ in Example 2). In more detail, based on the guessed model, a reduct of the (ground) program is created which does not contain default negation, and for which it can be checked whether the originally guessed model is a minimal model of the resulting reduct.

Given a program $P$ and a set $I$ of atoms, the *reduct* $P^I$ [32] is defined as

$$P^I = \{\mathcal{H} \leftarrow \mathcal{B}^+ : \mathcal{H} \leftarrow \mathcal{B}^+, \mathbf{not}\ \mathcal{B}^- \text{ in } \mathsf{ground}(P) \text{ such that } \mathcal{B}^- \cap I = \emptyset\}.$$

Intuitively, rules that contain an atom in $I$ (assumed to be true) in the negative body are removed. In the remaining rules, all negated atoms are removed.

The resulting program is positive, and for normal programs, we can determine all necessary consequences of this program using the following operator $T_P$:

$$T_P(I) = \{H \mid H \leftarrow \mathcal{B} \in P \text{ and } \mathcal{B} \subseteq I\}$$

Basically, all rule heads are collected whose body atoms are true given $I$.

This operator can be iterated as follows, starting with $T_P \uparrow 0$, thus allowing to compute the deductive closure $Cn$ of such a positive normal program $P$:

$$T_P \uparrow 0 = \emptyset \qquad T_P \uparrow (n+1) = T_P(T_P \uparrow n) \qquad Cn(P) = T_P \uparrow \omega = \bigcup_n T_P \uparrow n$$

Then, a set of atoms $X$ is an *answer set* of program $P$ iff $Cn(P^X) = X$.

For normal programs without negation, this closure in fact amounts to computing all necessary consequences of the given program. For example, for the ground program in Example 2 which does not contain default negation, clearly $\{\mathit{SSN\_OK}(\mathit{Bob}), \mathit{hasSSN}(\mathit{Bob}, 123)\}$ is this closure, hence its only answer set.

*Example 3.* Consider a normal program $P$ containing just two rules.

$$\mathit{HeartLeft}(\mathit{Bob}) \leftarrow \textbf{not } \mathit{HeartRight}(\mathit{Bob})$$
$$\mathit{HeartRight}(\mathit{Bob}) \leftarrow \textbf{not } \mathit{HeartLeft}(\mathit{Bob})$$

If we consider $M_1 = \{\mathit{HeartLeft}(\mathit{Bob}), \mathit{HeartRight}(\mathit{Bob})\}$, then $P^{M_1} = \{\}$, whose closure is $\emptyset$, hence $M_1$ is not an answer set. If we consider $M_2 = \{\}$, then $P^{M_2} = \{\mathit{HeartLeft}(\mathit{Bob}) \leftarrow, \mathit{HeartRight}(\mathit{Bob}) \leftarrow\}$ and the closure is $\{\mathit{HeartLeft}(\mathit{Bob}), \mathit{HeartRight}(\mathit{Bob})\}$, so $M_2$ is not an answer set either. Now, consider $M_3 = \{\mathit{HeartLeft}(\mathit{Bob})\}$. Then $P^{M_3} = \{\mathit{HeartLeft}(\mathit{Bob}) \leftarrow\}$, and $M_3$ is an answer set. The same is true for $M_4 = \{\mathit{HeartRight}(\mathit{Bob})\}$ by symmetry.

Indeed, in general, a program may have several answer sets, which corresponds to the idea that combinatorial problems may have several solutions. In particular, a program may also have no answer sets – take $\mathit{Person}(\mathit{Bob})$ together with rules (9) and (10). This is intended as we do not know the social security number of Bob, and in more general terms combinatorial problems may have no solution.

### 3.2 Well-Founded Semantics

The well-founded semantics [31] was developed around the same time as the answer set semantics, but focuses more on query answering over knowledge expressed with nonmonotonic rules. A central idea is to avoid computing entire models for a given program, but rather compute inferences in a top-down fashion, i.e., start with a query and compute only the part of a model necessary to obtain the answer. This is aligned with the ideas of Prolog [76], however unlike Prolog, the well-founded semantics is fully declarative and its major efficient implementation XSB Prolog [77] avoids undecidability caused by infinite loops while querying, and uses tabling to avoid unnecessary re-computation of already queried (intermediate) results.

Unlike the answer set semantics, the well-founded semantics is based on three-valued interpretations for which the set of truth values is extended by introducing a third truth value representing undefined. The basic idea behind this is to stay agnostic in situations where the knowledge encoded in a program permits a choice, such as between $HeartLeft(Bob)$ and $HeartRight(Bob)$ in Example 3, and rather leave both undefined. The benefit is that only a single well-founded model exists which can be efficiently computed in an iterative manner and for which corresponding querying procedures can be defined.

Formally, given a program $P$, a *three-valued interpretation* is a pair of sets of atoms $(T, F)$ with $T \cap F = \emptyset$, where elements in $T$ are mapped to true, elements in $F$ are mapped to false, and the remaining to undefined. Such interpretations can also be represented as $T \cup \textbf{not } F$. For example, using the latter notation, the well-founded model of the program in Example 3 can be represented as $\{\}$ corresponding to both $HeartLeft(Bob)$ and $HeartRight(Bob)$ being undefined, whereas the sets $\{HeartLeft(Bob), \textbf{not } HeartRight(Bob)\}$ and $\{\textbf{not } HeartLeft(Bob), HeartRight(Bob)\}$ correspond to the two answer sets in Example 3, respectively.

We can determine the truth value of a set of atoms involving undefined atoms by defining that, for an atom $A$ undefined w.r.t. some three-valued interpretation, **not** $A$ is again undefined and that for a conjunction of atoms, its truth value is the minimum of the truth values of the involved elements with respect to the order false $<$ undefined $<$ true.

Based on this, we can define the well-founded model and show how it can be computed. Here, we only do the latter and refer for the formal definition of the well-founded model to [31]. Essentially, the well-founded model can be computed by starting with an empty set (recall that this corresponds to everything is undefined), and iteratively add, based on the program, atoms that are necessarily true and necessarily false.

Regarding necessarily true information, the operator $T_P$ defined previously for computing the closure $Cn$ can be used, only now $I$ is a three-valued interpretation.

For negative information, so-called unfounded sets are used, that refer to a set of atoms that given a program $P$ and an interpretation $I$ can never become true (nor undefined). Formally, such a set $U$ is unfounded w.r.t. program $P$ and interpretation $I$ if each atom $A \in U$ satisfies the following condition. For each rule $A \leftarrow \mathcal{B}$ in $\mathsf{ground}(P)$ at least one of the following holds:

(Ui) Some literal in $\mathcal{B}$ is false in $I$.
(Uii) Some (non-negated) atom in $\mathcal{B}$ occurs in $U$.

The idea is that $A$ is unfounded if all rules with head $A$ either contain some false literal in the body or an atom which is also unfounded. For example, given $I = \emptyset$ and program $P$ composed of the two rules:

$$hasSSN(Bob, 123) \leftarrow isSSNOf(123, Bob)$$
$$isSSNOf(123, Bob) \leftarrow hasSSN(Bob, 123)$$

$\{hasSSN(Bob, 123), isSSNOf(123, Bob)\}$ is an unfounded set w.r.t. $P$ and $I$.

The union of all unfounded sets of $P$ w.r.t. $I$ is the greatest unfounded set, denoted $U_P(I)$ which together with $T_P$ can be used to define an operator $W_P$ for programs $P$ and interpretations $I$:

$$W_P(I) = T_P(I) \cup \textbf{not}\ U_P(I).$$

This operator can be iterated to obtain the well-founded model $M_{wf}$, i.e., all atoms that are necessarily true and false, respectively.

$$W_P \uparrow 0 = \emptyset \qquad W_P \uparrow (n{+}1) = W_P(W_P \uparrow n) \qquad M_{wf}(P) = W_P \uparrow \omega = \bigcup_n W_P \uparrow n$$

*Example 4.* Consider the following program $P$.

$$\begin{aligned} Person(Bob) &\leftarrow \\ hasSSN(Bob, 123) &\leftarrow \\ SSN\_OK(x) &\leftarrow hasSSN(x, y) \\ check(x) &\leftarrow Person(x), \textbf{not}\ SSN\_OK(x) \end{aligned}$$

We obtain:

$$\begin{aligned} &W_P \uparrow 0 = \emptyset \\ &W_P \uparrow 1 = \{Person(Bob), hasSSN(Bob, 123)\} \\ &W_P \uparrow 2 = W_P \uparrow 1 \cup \{SSN\_OK(Bob)\} \\ &W_P \uparrow 3 = W_P \uparrow 2 \cup \{\textbf{not}\ check(Bob)\} \end{aligned}$$

An alternative equivalent definition exists, called the alternating fixed-point [30], which computes this model based on the reduct of the program used for determining answer sets, and SLG resolution [20] provides a corresponding top-down procedure for the well-founded semantics implemented in XSB.

Regarding a comparison between the well-founded semantics and the answer set semantics, we note that in terms of computational complexity, the former is preferrable as the unique well-founded model can be computed in polynomial time (w.r.t. data complexity, i.e., only the size of the number of facts varies), whereas guessing and checking answer sets is at least in NP. In addition, when querying, we only require the part of the knowledge base that is relevant for the query, which aids efficiency in comparison to answer sets. For example, when querying for the medication of a specific patient, we certainly do not care about the medication of possibly thousands of other patients. Moreover, the well-founded model (for normal programs) always exists. On the other hand, if the considered problem is at least partially combinatorial, answer sets are clearly preferred. In fact, the answer set semantics is more expressive in general: take the two rules from Example 3 together with the following two rules.

$$\begin{aligned} LivingBeing(Bob) &\leftarrow HeartLeftBob) \\ LivingBeing(Bob) &\leftarrow HeartRight(Bob) \end{aligned}$$

Then, in the unique well-founded model everything is undefined, whereas both answer sets contain $LivingBeing(Bob)$. Thus, ultimately the choice between the depends on the intended application.

## 4 How to Integrate Ontologies and Rules?

Having reviewed the two formalisms, DL ontologies and nonmonotonic rules in detail, and their differing characteristics and benefits w.r.t. what kind of knowledge can be represented and reasoned with, the question arises what to do if we want to use the favorable characteristics of both simultaneously.

This question has been tackled in the literature and a plethora of different approaches has been presented. Discussing all these proposals here in detail would not be possible for the sheer amount of them, however, common characteristics and criteria have emerged along which these proposals have been defined, and we want to discuss these here to provide a better idea on the main considerations to take into account when providing such an integration.

Before we delve into this, let us look at a concrete, larger example that illustrates the benefits of such an integration with more detail.

*Example 5.* Recall the setting described in Example 1 on a customs service needing to assess incoming cargo for risks based on a variety of information. Figure 3 shows a DL ontology $\mathcal{O}$ and a set of nonmonotonic rules $\mathcal{P}$ containing part of such information that we explain in more detail.

The ontology $\mathcal{O}$ contains a classification of commodities based on their harmonized tariff information (HTS chapters, headings and codes)[5], a taxonomy of commodities (here exemplified using special kinds of tomatos), together with indications on tariff charges depending on the kind of product and their packaging, as well as a geographic classification, along with information about producers who are located in various countries.

The program $\mathcal{P}$ contains data on the shipments. Here, a shipment has several attributes: the country of origin, the commodity it contains, its importer and producer, exemplified by (partial) information about three shipments, $s_1$, $s_2$ and $s_3$. For importers, it also indicates if they have a history of transgressing the regulations.

Then there is a set of rules for determining what to inspect. The first rules provide information about importers, namely whether they are admissible (if they are not known to be registered transgressors), for which kind of products they are approved, and whether they are expeditable (combining the former two). The rules also allow us to associate a commodity with its country of origin (where it shipped from), and determining whether a shipment is compliant, i.e., if there is a match between the filed cargo codes and the actually carried commodities.

The final three rules serve the overall task to access all the information and assess whether some shipment should be inspected in full detail, partially, or not at

[5] This is adapted from https://hts.usitc.gov/.

$$* \; * \; * \; \mathcal{O} \; * \; * \; *$$

$\mathsf{Commodity} \equiv (\exists \mathsf{HTSCode}.\top)$

$\mathsf{CherryTomato} \equiv \exists \mathsf{HTSCode}.\{'07022'\}$

$\mathsf{GrapeTomato} \equiv \exists \mathsf{HTSCode}.\{'07021'\}$

$\mathsf{CherryTomato} \sqsubseteq \mathsf{Tomato}$

$\mathsf{CherryTomato} \sqcap \mathsf{Bulk} \equiv \exists \mathsf{TariffCharge}.\{'\$0'\}$

$\mathsf{GrapeTomato} \sqcap \mathsf{Bulk} \equiv \exists \mathsf{TariffCharge}.\{'\$40'\}$

$\mathsf{CherryTomato} \sqcap \mathsf{Prepackaged} \equiv \exists \mathsf{TariffCharge}.\{'\$50'\}$

$\mathsf{GrapeTomato} \sqcap \mathsf{Prepackaged} \equiv \exists \mathsf{TariffCharge}.\{'\$100'\}$

$\mathsf{EURegisteredProducer} \equiv \exists \mathsf{RegisteredProducer}.\mathsf{EUCountry}$

$\mathsf{LowRiskEUCommodity} \equiv (\exists \mathsf{ExpeditableImporter}.\top) \sqcap (\exists \mathsf{CommodCountry}.\mathsf{EUCountry})$

$\mathsf{EdibleVegetable} \equiv \exists \mathsf{HTSChapter}.\{'07'\}$

$\mathsf{Tomato} \equiv \exists \mathsf{HTSHeading}.\{'0702'\}$

$\mathsf{Tomato} \sqsubseteq \mathsf{EdibleVegetable}$

$\mathsf{GrapeTomato} \sqsubseteq \mathsf{Tomato}$

$\mathsf{CherryTomato} \sqcap \mathsf{GrapeTomato} \sqsubseteq \bot$

$\mathsf{Bulk} \sqcap \mathsf{Prepackaged} \sqsubseteq \bot$

$\mathsf{EUCountry}(\mathit{portugal})$ $\qquad$ $\mathsf{EUCountry}(\mathit{slovakia})$

$$* \; * \; * \; \mathcal{P} \; * \; * \; *$$

$\mathsf{ShpmtCommod}(s_1, c_1)$

$\mathsf{ShpmtImporter}(s_1, i_1)$

$\mathsf{ShpmtCommod}(s_2, c_2)$

$\mathsf{ShpmtImporter}(s_2, i_2)$

$\mathsf{ShpmtCountry}(s_2, \mathit{slovakia})$

$\mathsf{ShpmtCommod}(s_3, c_3)$

$\mathsf{ShpmtImporter}(s_3, i_3)$

$\mathsf{ShpmtCountry}(s_3, \mathit{portugal})$

$\mathsf{ShpmtCountry}(s_1, \mathit{portugal})$

$\mathsf{ShpmtDeclHTSCode}(s_1, \mathrm{h7022})$

$\mathsf{CherryTomato}(c_1) \quad \mathsf{Bulk}(c_1)$

$\mathsf{ShpmtDeclHTSCode}(s_2, \mathrm{h7021})$

$\mathsf{GrapeTomato}(c_2) \quad \mathsf{Prepackaged}(c_2)$

$\mathsf{RegisteredTransgressor}(i_1).$

$\mathsf{ShpmtDeclHTSCode}(s_3, \mathrm{h7022})$

$\mathsf{GrapeTomato}(c_3) \quad \mathsf{Bulk}(c_3)$

$\mathsf{ShpmtProducer}(s_3, p_1)$

$\mathsf{RegisteredProducer}(p_2, \mathit{slovakia})$

$\mathsf{AdmissibleImporter}(\mathbf{x}) \leftarrow \mathsf{ShpmtImporter}(\mathbf{y}, \mathbf{x}), \mathbf{not}\ \mathsf{RegisteredTransgressor}(\mathbf{x})$

$\mathsf{ApprovedImporterOf}(i_2, \mathbf{x}) \leftarrow \mathsf{EdibleVegetable}(\mathbf{x})$

$\mathsf{ApprovedImporterOf}(i_3, \mathbf{x}) \leftarrow \mathsf{GrapeTomato}(\mathbf{x})$

$\mathsf{ExpeditableImporter}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtImporter}(\mathbf{z}, \mathbf{y}),$
$\qquad\qquad \mathsf{AdmissibleImporter}(\mathbf{y}), \mathsf{ApprovedImporterOf}(\mathbf{y}, \mathbf{x})$

$\mathsf{CommodCountry}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtCountry}(\mathbf{z}, \mathbf{y})$

$\mathsf{CompliantShpmt}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{HTSCode}(\mathbf{y}, \mathbf{z}), \mathsf{ShpmtDeclHTSCode}(\mathbf{x}, \mathbf{z})$

$\mathsf{PartialInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{LowRiskEUCommodity}(\mathbf{y})$

$\mathsf{FullInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{CompliantShpmt}(\mathbf{x})$

$\mathsf{FullInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{Tomato}(\mathbf{y}), \mathsf{ShpmtCountry}(\mathbf{x}, \mathit{slovakia})$

**Fig. 3.** Ontology $\mathcal{O}$ and nonmonotonic rules $\mathcal{P}$ for cargo assessment.

all. In particular, at least a partial inspection is required whenever the commodity is not a LowRiskEUCommodity based on inferences in the DL part, which n turn requires assessing further information in the rules, namely CommodCountry and ExpeditableImporter. A full inspection is required if a shipment is not compliant or if some suspicious cargo is observed, in this case tomatoes from slovakia (you may imagine for the sake of the example that we are in the winter period).

Note that the example indeed utilizes the features of rules and ontologies: for example, exceptions to the partial inspections can be expressed, but at the same time, taxonomic and non-closed knowledge is used, e.g., some shipment may in fact originate from the EU, but this information is just not available.

To be able to obtain the desired conclusions from the combination of the knowledge bases written in these two formalisms, we need to find ways to integrate them. As already mentioned, many proposals exist in the literature, defined along the lines of guiding principles which we now want to discuss. In the course of this discussion, often, these principles are inspired in their formulation by [62], where they were used for arguing in favor of a specific approach. Here, in particular when there are several options w.r.t. a certain characteristic, it is our stance not to argue in favor of a certain solution, but rather discuss their corresponding advantages.

*Faithfulness* The first desirable property we want to discuss is Faithfulness, i.e., the idea that the integration of DLs and nonmonotonic rules should preserve the semantics of both its base formalisms. In other words, the addition of rules to a DL should not change the semantics of the DL and vice-versa. In particular, when either of the two components is empty, the semantics should simply be the one of the base formalism of the non-empty component. This is beneficial for two reasons. First, it eases its adoption for knowledge engineers knowledgable in (at least) one of the base formalisms, that want to augment an ontology with rules or vice-versa. Second, and maybe even more important, this is crucial to facilitate re-using existing state of the art reasoners for each of the components, reducing considerably the necessary effort to provide implementations. Thus, without going into details of an actual faithful integration, for reasoning with the knowledge bases presented in Example 5, it would in principle suffice to choose a DL reasoner appropriate for the expressiveness used in the DL and an ASP solver (or XSB depending on the desired rule semantics) and determine an interface between the two to obtain all desired inferences.

*Tightness* The characteristic of tightness relates to the structure/topology of the integration in the sense that whether conclusions resulting from one of the formalisms can be used to derive further conclusions in the other. Clearly, there are several possible solutions. One option is to layer one formalism on top of the other, in the sense that conclusions of one approach can be used by the other, but not vice-versa. Such a solution is certainly easier to handle on the technical level, since, assuming a faithful integration, this way, we can simply first compute the conclusions within the formalism on the lower level, and pass these to the formalism on the upper level to compute the conclusions there. On the other hand, a tight integration requires that neither of the two formalisms is layered on top of the other, rather, both the DL and the rule component should be able to contribute to the consequences of the other component. This is technically more advanced, but has the advantage that it can easily cover examples such as the one presented in Example 5, where, e.g., LowRiskEUCommodity is used in the rules, but derived in the ontology, which in turn requires inferences from the rules. Of course, in applications where such tight integration is not necessary, a layered solution suffices. On the other hand, a tight integration is not subject to problems when later introducing transfer of inferences from one component to the other that was previously not present. Finally, there are also intermediate solutions in which inferences can be passed between both components of the integration, but under some certain restrictions.

*DL Reasoning View* This characteristic refers to what kind of inferences are passed from the DL component to the rule component of the integration. Clearly, for layered approaches in which rules are on the lower layer, this characteristic is irrelevant. However, the vast majority of approaches actually do pass conclusions from the DL component to the rule component. The essential question is, given a DL ontology, do we consider inferences on the level of individual models or on the level of consequences, i.e., truth in all models? In other words, are the inferences we pass from the ontology to the rules model-based or consequence-based, i.e., is the integration a world-centric or entailment-centric approach (as alternatively termed in [56])? The benefit of model-based inferences is that they allow for a higher expressiveness in the sense that possible alternatives present in the ontology can be passed to the rules resulting in more inferences there as well. On the downside, such alternatives based on different models may not have strong support as a conclusion obtained from the ontology. For consequence-based approaches, the situation is exactly the converse. Any inference is true in all models, but this limits the extent to which inferred information is passed from the ontology to the rules. This is ultimately actually similar to the idea of brave vs. cautious reasoning in logics, i.e., truth in one or all models, and which one of the two views is adapted depends on the concrete application at hand. For Example 5, this depends on whether we want to impose inspections based on something possibly being true or an entailment, i.e., being true in all models. For example, LowRiskEUCommodity should probably only be considered under consequences, as discarding a partial inspection based on the fact that the commodity in question may represent a lower risk, is probably not a good idea.

*Flexibility* This characteristic deals with the question whether the same predicate can be viewed under both the open and closed world assumption. The central idea is that a flexible approach allows us to enrich a DL ontology with non-monotonic consequences from the rules, and, conversely, to enrich the rules with the capabilities of ontology reasoning described by a DL ontology. This allows us to distinguish between approaches that fix to which predicates either OWA or CWA is applied and those that do not. Again, a flexible approach is commonly more expressive, possibly at the expense of requiring a technically more advanced integration, since separate languages for the two components facilitate the re-use of existing semantics for the individual components. Yet, this may require some additional effort to ensure that corresponding predicates (between the ontology and the rules) are appropriately synchronized. In the case of Example 5, an approach capturing it as is needs to be flexible as the predicates appear simultaneously in both components.

*Decidability and Complexity* Finally, a formalism that can be used in applications should be at least decidable, and preferably of low worst-case complexity. Given the huge amount of data (on the Web), it is clearly preferable to have a system that is not only decidable but also computationally as efficient as possible. Still, this needs to take into account the tradeoff between expressiveness of the formalism and complexity of reasoning, and in certain situations, the required expressiveness is more important. In any case, both base formalisms come with established methods to ensure decidability, and this should preferably be maintained in their integration. The exact computational complexity for desired reasoning tasks then depends on the expressiveness and semantics applied in the two base formalisms and which of the beforementioned characteristics have been adopted.

## 5 Concrete Integrations

Having reviewed the main guidelines/criteria along which the many existing approaches for integrating ontologies and rules have been developed, in this section, we want to give an overview of some of the more prominent such proposals. While the question of prominence to some extent certainly is in the eye of the beholder, and though we cannot discuss all existing approaches here in more detail, we believe that the provided selection covers different aspects w.r.t. the previously mentioned criteria and more details on the remaining approaches can be obtained from pointers to the literature.

### 5.1 $\mathcal{DL}$+log

One of the first combinations of non-monotonic rules and ontologies is called r-hybrid knowledge bases [67], which subseqently has been extended to $\mathcal{DL}$+log [68]. $\mathcal{DL}$+log combines disjunctive Datalog (consisting of rules of the form (8)) with an arbitrary (decidable) DL.

An essential idea is to separate the admitted predicates into DL predicates and non-DL predicates, i.e., those that can appear in the DL part (and in the rules), and those that only appear in the rules. A knowledge base $\mathcal{K}$ in this formalism consists of an ontology $\mathcal{O}$ and a set of rules $\mathcal{P}$, where each rule $r$ is of the following form:

$$p_1(\boldsymbol{x}_1) \vee \ldots \vee p_n(\boldsymbol{x}_n) \leftarrow r_1(\boldsymbol{y}_1), \ldots, r_m(\boldsymbol{y}_m), s_1(\boldsymbol{z}_1), \ldots, s_k(\boldsymbol{z}_k),$$
$$\mathbf{not}\ u_1(\boldsymbol{w}_1), \ldots, \mathbf{not}\ u_h(\boldsymbol{w}_h)$$

such that $n \geq 0$, $m \geq 0$, $k \geq 0$, and $h \geq 0$, the $\boldsymbol{x}_i$, $\boldsymbol{y}_i$, $\boldsymbol{z}_i$, and $\boldsymbol{w}_i$ are tuples of variables and constants, each $p_i$ a DL or a non-DL predicate, each $s_i$ a DL predicate, and each $r_i$ and $u_i$ a non-DL predicate. Additionally, each variable occurring in a rule must appear in some $\boldsymbol{y}_i$ or $\boldsymbol{z}_i$ (rule safety – similar to what we have seen in Section 3) and every variable appearing in some $\boldsymbol{x}_i$ of $r$ must appear in at least one of the $\boldsymbol{y}_i$ (weak safety). The latter notion is weaker/less restrictive than DL-safety [63] applied for r-hybrid knowledge bases, which requires that every variable (and not just those in $\boldsymbol{x}_i$) in the rule occurs in at least one $\boldsymbol{y}_i$, since there may exist variables that only appear in a DL-atom in the body of a rule. For example, we may assume that, in Example 5, data on registered producers is also stored in the ontology (in the ABox), i.e., EURegisteredProducer is a DL-predicate, and adapt the DL axiom on EURegisteredProducer as a rule:

$$\mathsf{EURegisteredProducer}(\mathbf{x}) \leftarrow \mathsf{RegisteredProducer}(\mathbf{x}, \mathbf{y}), \mathsf{EUCountry}(\mathbf{y}) \tag{11}$$

Then, this rule is weakly-safe, but not DL-safe, as, according to our assumptions, $\mathbf{x}$ does not occur in the rule body in an atom built over a non-DL predicate (nor does $\mathbf{y}$). This is interesting as it allows one to pose arbitrary conjunctive queries over DL predidates to the DL component, for which there are known algorithms for many DLs [57]. The standard rule safety nevertheless ensures that there is no variable only appearing in a default negated atom.

The main idea of the semantics for $\mathcal{DL}$+log [68] is based on the definition of a projection, similar in spirit to the idea of the reduct for answer sets, which intuitively simplifies the program by evaluating the DL-atoms. Basically, given an interpretation $I$, rules with DL-atoms that conflict with $I$ are deleted and the remaining DL-atoms are omitted. The resulting program is free of DL-atoms, and $\mathcal{I}$ is a model of $\mathcal{K}$ if $\mathcal{I}$ is a model of $\mathcal{O}$, and $\mathcal{I}$ restricted to the non-DL predicates is an answer set of the projected program. This way, DL atoms are evaluated under the open world assumption in the spirit of conjunctive queries to the DL part, whereas non-DL atoms are evaluated under the closed world assumption.

*Example 6.* Consider a simple example composed of rule (11) together with two ABox assertions $\mathsf{RegisteredProducer}(s_4, GB)$, $\mathsf{CountryIE}(GB)$ and an axiom

$$\mathsf{CountryIE} \sqsubseteq \mathsf{EUCountry} \sqcup \mathsf{NonEUCountry}$$

representing that any country in Europe is either in the European Union or not, and that $s_4$ is a registered producer in Great Britain, which is a country in Europe. Then there are two models:

$$\begin{aligned} M_1 = &\{\mathsf{RegisteredProducer}(s_4, GB), \mathsf{CountryIE}(GB), \mathsf{EUCountry}(GB),\\ &\mathsf{EURegisteredProducer}(s_4)\}\\ M_2 = &\{\mathsf{RegisteredProducer}(s_4, GB), \mathsf{CountryIE}(GB), \mathsf{NonEUCountry}(GB)\} \end{aligned}$$

Notably, $M_1$ and $M_2$ correspond to the essential two models of $\mathcal{O}$, and, for $M_2$, the evaluation of DL-atoms removes all instances of (11).

This then allows the definition for reasoning algorithms along the idea of guessing and checking interpretations based on these projections, and it has been shown that, in combination with $\mathcal{O}$ in DL-Lite, the DL underlying OWL 2 QL, the data complexity does not increase with respect to the data complexity of the rule component alone.

The semantics is faithful, tight, and decidable (due to the safety restrictions), but it is not flexible, as non-monotonic reasoning is by definition restricted to non-DL atoms, i.e., no default reasoning over information appearing in the DL is possible. Finally, we note that this approach is indeed model-based, since it suffices that a considered interpretation $\mathcal{I}$ models $\mathcal{O}$, without considering entailment, i.e., truth in all models of $\mathcal{O}$.

## 5.2 dl-programs

Another important approach is called description logic programs (dl-programs) [27]. Here, a knowledge base $\mathcal{K}$ consists of a DL ontology $\mathcal{O}$ and a program $\mathcal{P}$ containing (non-disjunctive) dl-rules of the form

$$H \leftarrow A_1, \leftarrow A_n, \textbf{not } B_1, \ldots, \textbf{not } B_m$$

where $H$ is a first-order atom,[6] and all $A_i$ and $B_j$ are first-order atoms or special dl-atoms to query the ontology. Similarly to $\mathcal{DL}$+log, the set of predicates is divided into disjoint sets of DL predicates, which only appear in dl-atoms and in $\mathcal{O}$, and non-DL predicates, that only appear in the rules. Such dl-atoms are meant to serve as interfaces between the ontology and the rules, allowing that information derivable in the DL can be queried for in the rules, while information in the rules can be passed to the DL in the course of this process.

As a simple example, consider that the rules contain a fact $\mathtt{p}(\mathtt{Bob})$, representing that $\mathtt{Bob}$ is a person, together with ontology axioms (5) and (6), from which we can derive that every person is a vertebrate. Note that the ontology and the rules use a different predicate to represent the concept of person. Then, a dl-atom $DL[\mathtt{Person} \uplus \mathtt{p}; \mathtt{Vertebrate}](\mathtt{Bob})$ represents a query for $\mathtt{Vertebrate}(\mathtt{Bob})$ where $\mathtt{Person}$ in the ontology is enriched with the inferences for $\mathtt{p}$ from the rules.

More formally, such dl-atoms are of the form

$$DL[S_1 \; op_1 \; p_1, \ldots, S_l \; op_l \; p_l; Q](\boldsymbol{t})$$

where $S_i$ are DL predicates, $p_i$ are non-DL predicates, $op_i \in \{\uplus, \cup\!\!\!\!-, \cap\!\!\!\!-\}$, $Q$ is an $n$-ary DL predicate, and $\boldsymbol{t}$ a vector of $n$ terms. Such dl-atoms are used to query the ontology for $Q(\boldsymbol{t})$ where certain ontology predicates $S_i$ are altered by information derived in the rules. Intuitively, $\uplus$ is used to augment $S_i$ with the derived knowledge from $p_i$; $\cup\!\!\!\!-$ augments $\neg S_i$ with the derived knowledge from $p_i$ and $\cap\!\!\!\!-$ augments $\neg S$ with what is not derived in $p_i$. As queries, concept inclusions, negations of concept inclusions, concept and role assertions, and equalities and inequalities are allowed.

[6] Classical negation is also allowed, but we simplify here for the sake of presentation.

It should be noted that the transfer from the rules to the ontology is limited though, in the sense that it is not stored. This means that if a certain piece of information from the rules is to be used for each dl-query, then it has to be added explicitly in every dl-atom.

*Example 7.* The rule from Example 5 for PartialInspection can be adapted as:

$$\mathsf{PartialInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}),$$
$$\mathbf{not}\ DL[\mathsf{ExpeditableImporter} \uplus \mathsf{e}, \mathsf{CommodCountry} \uplus \mathsf{c}; \mathsf{LowRiskEUCommodity}](\mathbf{y})$$

where e and c are non-DL predicates which replace ExpeditableImporter and CommodCountry in all the rules. In fact, to make this fully correct with the idea of dl-programs, all predicates that appear, in Example 5, in the ontology and in the rules need to be duplicated. This may look cumbersome at first glance, but it allows one to use the two components in a modular fashion which facilitates implementations, and reduces transfer of knowledge to what is necessary aiding efficiency.

Two different answer set semantics are defined slightly varying on how dl-atoms are preprocessed in the reduct. Both semantics may however create non-minimal answer sets which is why it is recommended to either use canonical answer sets based on loop formulas [80] or not to use the operator $\cap\!\!\!-$ at all.

In addition, a corresponding well-founded semantics is defined for dl-programs [26] omitting the operator $\cap\!\!\!-$, which builds on an extension of the unfounded sets construction to such dl-programs.

The approach itself is faithful for both versions, the answer set semantics and the well-founded semantics. The interaction with the DL component is consequence-based as the evaluation of queries in the dl-atoms is realized based on entailments, i.e., truth in all models of $\mathcal{O}$. Due to the specific interfaces, the dl-atoms, this approach is tight and flexible to a limited extent: rules cannot derive new facts about DL predicates, they can only pose conditional queries to the DL, although the operators in dl-atoms provide means to transfer knowledge temporarily. Moreover, we can never derive nonmonotonic consequences for DL predicates directly. Rather, we have to use the interfaces, which may appear under default negation. It has also been shown that dl-programs are decidable provided the considered DL is decidable. For the answer set semantics, in general the computational complexity increases that of the individual components (depending on the DL used and the kind of rules permitted), whereas for the well-founded semantics this can commonly be avoided. In particular for polynomial DLs, such as the tractable OWL profiles, dl-programs also maintain a polynomial data complexity. While the modular solution limits the approach in terms of tightness and flexibility, it facilitates its implementation. A prototype for both semantics has been defined that utilizes RACER [37] for DL reasoning and DLV [53, 3] for the rule processing (where the well-founded semantics is implemented based on the alternating fixpoint). This has been subsequently generalized in hex-programs [24, 66] where interfaces in the line of dl-atoms can be created to arbitrary external sources, thus further widening the applicability.

## 5.3 Hybrid MKNF

Hybrid MKNF knowledge bases [62] build on the logic of minimal knowledge and negation as failure (MKNF) [54], which corresponds to first-order logic extended by two modal operators **K** and **not**, that allow us to express that something is known and not known, respectively. Hybrid MKNF KBs essentially consist of two components: a first-order theory, in particular a DL ontology $\mathcal{O}$ (translatable into first-order logic), and a finite set of rules (similar to rules in ASP) over so-called modal atoms of the form

$$\mathbf{K}H_1 \vee \ldots \vee \mathbf{K}H_l \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$$

where the $H_i$, $A_j$, and $B_k$ are first-order formulas. The essential idea is that such a rule is to be read as "If all $A_j$ are known to hold (in the sense of truth in all models), and all $B_k$ are not known to hold, then one of the $H_i$ is known to hold" (i.e., one of the $H_i$ is true in all models). As modal operators are not admitted in the ontology $\mathcal{O}$, reasoning can still be applied within $\mathcal{O}$ on a per model basis as intended. Admitting, in the rules, first-order formulas within the scope of modal operators raises the expressivity, and allows, in such modal atoms, conjunctive queries to the ontology.

In fact, even more general MKNF$^{+}$ knowledge bases are considered in which rules may contain atoms not in scope of any modal operator. These can however be transformed into ordinary modal atoms provided the considered DL language is expressive enough to encode these first-order atoms with an equivalence to a new predicate used then in the rules in each such case.

Atoms are again divided into DL atoms and non-DL atoms, and for decidability, it is required that DL-safety holds, i.e., all variables occur in at least one non-DL atom, and that reasoning in the DL language together with the generalized atoms is decidable (the latter condition simplifies if no first-order formulas occur in the rules). Note that, due to DL-safety, rules such as (11) cannot be used, but this can be amended using a conjunctive query in a modal atom:

$$\mathbf{K}\mathsf{EURegisteredProducer}(\mathbf{x}) \leftarrow \mathbf{K}[\exists y.(\mathsf{RegisteredProducer}(\mathbf{x}, \mathbf{y}), \wedge \mathsf{EUCountry}(\mathbf{y}))]$$

The semantics of Hybrid MKNF knowledge bases is given by a translation into an MKNF formula, i.e., a formula over first-order logic extended with two modal operators **K** and **not**. This translation, essentially, conjoins all rules with the first-order translation of $\mathcal{O}$ within scope of a single modal operator **K**. The (integrating) semantics for such formulas is defined based on a model notion over sets of interpretations. Such models contain all interpretations that model a given formula, minimizing what necessarily must be known to hold, in the sense that a larger set of models contains less atoms that are true in all models, hence the name minimal knowledge (and negation as failure).

*Example 8.* Consider a simple example adapted from Example 5 containing just $\mathsf{CherryTomato} \sqsubseteq \mathsf{Tomato}$ and $\mathbf{K}\mathsf{Tomato}(o_1) \leftarrow$. The corresponding formula in MKNF is $\mathbf{K}(\forall x \mathsf{CherryTomato}(\mathbf{x}) \rightarrow \mathsf{Tomato}(\mathbf{x})) \wedge \mathbf{K}\mathsf{Tomato}(o_1)$. Then, $M_1 =$

$\{\{\mathsf{Tomato}(o_1)\}, \{\mathsf{Tomato}(o_1), \mathsf{CherryTomato}(o_1)\}\}$ contains sets that model the formula. So does $M_2 = \{\{\mathsf{Tomato}(o_1), \mathsf{CherryTomato}(o_1)\}\}$, but $M_2$ is not the maximal such set of sets. In fact, in $M_2$, $\mathsf{CherryTomato}(o_1)$ is true in all models, i.e., according to this model, $\mathbf{K}\mathsf{CherryTomato}(o_1)$ holds, which clearly should not be an inference of the given knowledge base.

Since such a model representation is cumbersome and in general infinite, for reasoning, a finite representation based on modal atoms is provided that indicates which modal atoms occurring in the program are true and false. This then allows one to use algorithms whose ideas are closely aligned with that of answer sets – guess the true modal atoms and check whether certain conditions are verified, namely that the result is a minimal model for the formula. For Example 8, this amounts to determining that $\mathbf{K}\mathsf{Tomato}(o_1)$ is true.

A well-founded semantics based on alternating fixpoints is defined in [48] for hybrid MKNF knowledge bases, where no disjunction is allowed in the rule heads and only ordinary atoms are permitted in rules, i.e., no first-order formulas.

Due to the seemless embedding in the underlying unifying formalism, the approach is naturally flexible and tight. This allows us for example to capture Example 5, by simply introducing the modal **K** operators in all the rules (the default **not** is already present). It also is faithful with the respective underlying base formalisms and decidable, provided the required restrictions are satisfied. In terms of complexity, the semantics based on answer sets does in general increase the data complexity of its constituents, only under considerable restrictions on the rules, this can be avoided. For the well-founded semantics on the other hand, and similar to dl-programs, the complexity reduces in comparison, and for polynomial DLs, polynomial data complexity can be ensured for the integrated formalism (for computing the model as well as answering safe queries). Moreover, the approach is consequence-based as DL atoms appear in scope of modal operators, hence they are verified to be known, i.e., true in all models (of $\mathcal{O}$).

The approach is indeed very general, and MKNF$^+$ knowledge bases allow us to cover many approaches in the literature [62], including the ones discussed so far, in the sense that $\mathcal{DL}$+log KBs and dl-programs without the operator $\Cap$ can be encoded into an equisatisfiable hybrid MKNF knowledge base.

In the latter case, a complementary formal result [28] shows that DL-safe and ground hybrid MKNF KBs can be embedded into dl-programs essentially by establishing the necessary transfer of information from rules to ontologies in every single DL-atom. This confirms that Example 5 which can be straightforwardly handled in hybrid MKNF, is also capturable in the case of dl-programs. The general idea is to just introduce one new auxiliary predicate for each DL atom, basically creating a program and a DL version of each such predicate, and then use dl-atoms in the bodies of rules (instead of DL atoms), to query the ontology, where, in the dl-atoms, the inferred information for all such DL atoms from the rules is passed to the corresponding DL component each time.

### 5.4 Resilient Logic Programs

Resilient Logic Programs (RLPs) [56] have been recently introduced with the aim to overcome a limitation of the existing approaches of integrations of ontologies and rules, namely the fact that, for obtaining inferences from the ontology, they use either model-based reasoning (e.g., $\mathcal{DL}$+log) or consequence-based reasoning (e.g., dl-programs and Hybrid MKNF), but not both. There are however problems where this is not sufficient, and the authors argue that integration solutions should be resilient in various scenarios.

As an example, consider that we are given a set of nodes, and we want to determine a directed graph $G$ such that removing one arbitrary node from $G$ will always result in a strongly connected graph (i.e., every vertex is reachable from every other vertex). In this case, the choice of which node is removed can be modeled in the ontology under the model-based view (one model per possible removed node), whereas we can use the rules to verify whether the corresponding resulting graph is strongly connected. However, the reachability relation involved in this check varies for different chosen nodes, which requires a universal quantification over the choices in the ontology (aligned with the consequence-based view).

A resilient logic program then is a tuple $\Pi = (\mathcal{P}, \mathcal{O}, \Sigma_{out}, \Sigma_{owa}, \Sigma_{re})$ consisting of a program $\mathcal{P}$, an ontology/first-order theory $\mathcal{O}$, and a distinct partition of the predicates occurring in $\mathcal{P}$ and $\mathcal{O}$, into output predicates $\Sigma_{out}$, open predicates $\Sigma_{owa}$, and response predicates $\Sigma_{re}$, such that response predicates are not allowed in $\mathcal{O}$, and where output predicates and response predicates are closed.

The semantics is defined in a way that can be viewed as a negotiation between $\mathcal{P}$ and $\mathcal{O}$. An answer set $I$ over $\Sigma_{out}$ needs to be determined that a) can be extended into a model of $\mathcal{O}$ by interpreting the predidates in $\Sigma_{owa}$, and b) no matter how $I$ is extended into a model of $\mathcal{O}$, there always is a corresponding interpretation of the response predicates, which together with $I$ is justified by $\mathcal{P}$ (in the sense of minimality/support by a rule as argued before, e.g., for answer set programs). This semantics uses a reduct inspired by [67, 9], which handles default negated atoms in the rules as usual, but in addition also treats atoms based on open predicates in a similar fashion.

*Example 9.* Consider the following formalization of the graph problem taken from [56], where $\mathcal{O}$ is represented as a first-order theory.
For nodes $n_1, \ldots, n_k$, let $\Pi = (\mathcal{P}, \mathcal{O}, \Sigma_{out} = \{V, E\}, \Sigma_{owa} = \{in, out\}, \Sigma_{re} = \{\bar{E}, R\})$, where

$$\begin{aligned}
\mathcal{O} = \{ & \exists x.out(x) \qquad \forall x.(V(x) \rightarrow (in(x) \lor out(x))), \\
& \forall x.(V(x) \rightarrow (\neg in(x) \lor \neg out(x))), \\
& \forall x \forall y.((out(x) \land out(y)) \rightarrow x = y)\} \\
\mathcal{P} = \{ & V(n_1), \ldots V(n_k), \qquad E(x,y) \lor \bar{E}(x,y) \leftarrow V(x), V(y), \\
& R(x,z) \leftarrow R(x,y), R(y,z), \\
& R(x,y) \leftarrow E(x,y), \mathbf{not}\ out(x), \mathbf{not}\ out(y), \\
& \leftarrow V(x), V(y), x \neq y, \mathbf{not}\ out(x), \mathbf{not}\ out(y), \mathbf{not}\ R(x,y)\}
\end{aligned}$$

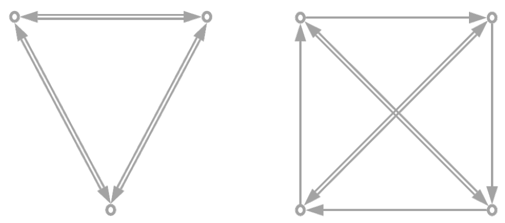


**Fig. 4.** A graph representation of the unique solution for Example 9 for 3 nodes, and one possible solution for 4 nodes.

Essentially, $\mathcal{O}$ allows us to choose exactly on vertex that is removed (out), whereas $\mathcal{P}$ provides possible directed edges between the given vertices ($E(x, y)$ corresponding to a chosen edge for vertices $x$ and $y$, $\bar{E}(x, y)$ representing one not chosen), determines reachability (using $R(x, y)$), and a constraint checking that any remaining vertex can reach any other remaining vertex. The semantics of RLPs then only admits the desired solutions in which removing any single node results in a strongly connected graph. Figure 4 shows a representation of the only possible such graph with three nodes and one possible solution with four nodes. It is easy to see that removing any of the directed edges will result in a graph for which the required condition is no longer satisfied.

A more extensive example is presented in [56], where a company wants to process a fixed amount of customer orders per day. The exact configuration of the orders is not known in advance, and the objective is to determine those services to offer so that independently of the actual configurations of the orders, the tasks can be assigned to employees so that each task is completed by the end of the day. Here, the offered services are captured by the output predicates, possible configurations are modelled in the first-order theory, and the answer sets correspond to the viable schedules in which tasks are completed in time.

Decidability in RLPs is achieved by ensuring DL-safety [63], here w.r.t. output and response predicates, and requiring that for the (DL) theory, satisfiability under closed predicates must be decidable.

It is shown that disjunctive programs can be embedded into RLPs. Also, under some additional restrictions, namely limiting default negation for response predicates and restricting theories to correspond to positive disjunctive rules, RLPs can be translated into disjunctive ASP which allows for the usage of state-of-the-art ASP solvers when reasoning. A reduction to $\exists\forall\exists$-quantified Boolean formulas is given, which shows that the computational complexity is higher than that of answer sets, which is also confirmed when using concrete DLs of different expressiveness. In the context of these concrete DLs, a relaxed safeness condition is also provided that admits safety with respect to unknown individuals as long as the number of these individuals is limited.

The approach is faithful w.r.t. the base formalisms and decidable under the imposed restrictions. It also is tight as the flow of information is possible in both directions. Similar to $\mathcal{DL}$+log, the approach is not flexible as open predicates are interpreted under the open world assumption.

## 6 NoHR - Querying Ontologies and Rules

In this section, we discuss NoHR[7] (Nova Hybrid Reasoner), a tool for querying combinations of DL ontologies and nonmonotonic rules. It is based on the well-founded semantics for Hybrid MKNF knowledge bases [48], due to its lower computational complexity and amenability to top-down querying without computing the entire model, which is important in the face of huge amounts of data. Indeed, rather than computing the well-founded model for this integration along the ideas presented in Section 3.2, queries are evaluated based on $\mathbf{SLG(\mathcal{O})}$, as defined in [2]. This procedure extends SLG resolution with tabling [20] with an *oracle* to $\mathcal{O}$ that handles ground queries to the DL-part of $\mathcal{K}$ by returning (possibly empty) sets of atoms that, together with $\mathcal{O}$ and information already proven true, allows us to derive the queried atom. We refer to [2] for the full account of $\mathbf{SLG(\mathcal{O})}$ and present the idea in the following example.

*Example 10.* Consider again Example 5 and the query $\mathsf{PartialInspection}(\mathbf{x})$. Then, the query procedure would find the rule whose head matches the queried atom and proceed by querying for the respective body elements $\mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y})$ and **not** $\mathsf{LowRiskEUCommodity}(\mathbf{y})$. We find $\mathsf{ShpmtCommod}(s_1, c_1)$, then the remaining query is **not** $\mathsf{LowRiskEUCommodity}(c_1)$, which is verified with a query $\mathsf{LowRiskEUCommodity}(c_1)$. Now, $\mathsf{LowRiskEUCommodity}$ is a DL-predicate and this can be handled by the oracle to $\mathcal{O}$. In fact, $\mathsf{LowRiskEUCommodity}(c_1)$ can be inferred from $\mathcal{O}$ if we find $\mathsf{ExpeditableImporter}(c_1, \mathbf{x})$ and $\mathsf{CommodCountry}(c_1, \mathbf{y})$ and $\mathsf{EUCountry}(\mathbf{y})$. Querying for $\mathsf{ExpeditableImporter}(c_1, \mathbf{x})$ in particular results in a query $\mathsf{AdmissibleImporter}(i_1)$ which fails as $i_1$ is a registered transgressor. Therefore, $\mathsf{LowRiskEUCommodity}(c_1)$ eventually fails, and one answer for the initial query is a partial inspection is required for $s_1$.

While this idea works from a semantic point of view, it leaves open the question of efficiency, as in general there are exponentially many such sets of atoms that together with the ontology allow us to derive the queried atom. The solution applied in NoHR is to transform relevant axioms/logical consequences of the considered ontology into rules, and then take advantage of a rule reasoner to ensure that only polynomially many answers are returned. It has been shown that this process preserves the semantics for the different permitted ontology fragments [43, 21, 55]. In addition, due to this reasoning approach, DL safety can be relaxed to rule safety, as DL atoms now refer to the rules resulting from the translation.

[7] http://nohr.di.fct.unl.pt

On the technical side, NoHR[8] is developed as a plug-in for the ontology editor Protégé 5.X,[9] – in fact, the first hybrid reasoner of its kind for Protégé – but it is also available as a library, allowing for its integration within other environments and applications. It supports ontologies written in any of the three tractable OWL 2 Profiles, and for those combining the constructors permitted in these profiles. Its implementation combines the capabilities of the DL reasoners ELK [47] (for the OWL 2 EL profile), and HermiT [34] and Konclude [75] (for combinations of constructors of different profiles – for OWL 2 QL and RL no DL reasoner is used, rather direct translations into rules are applied) with the rule engine XSB Prolog[10]. XSB comes with support for a vast number of standard built-in Prolog predicates, including numeric predicates and comparisons, and ensures termination of query answering.

NoHR is also robust w.r.t. inconsistencies between the ontology and the rules, which is important as knowledge from different sources may indeed be contradictory on some parts. While this would commonly render the system useless as anything can be derived from an inconsistent knowledge base, a paraconsistent approach is adopted, in which certain parts may be inconsistent – and querying for them with NoHR will reveal that – but other inferences that are not related to such an inconsistency can be inferred as if the inconistency was not present (see, e.g., [44] for more details). This proves indeed beneficial as inconsistent data on one shipment should not impact on determining whether to inspect another.

NoHR also provides native support for Relational Databases which is encoded through the concept of mappings.[11] Essentially, mappings are used to create predicates that are populated with the result set obtained from queries to external databases, which also allows one to consult tables from different databases. A mapping for predicate $p$ is a triple $\langle p, db, q\rangle$ where the result set from $db$ for the query $q$ (defined over $db$) is mapped to the predicate $p$. The connection to database systems is realized using ODBC drivers, thus allowing the integration of NoHR with all major database management systems.

In the following, we describe the system architecture of the Protégé plugin NoHR v4.0 as shown in Fig. 5 and discuss several features of its implementation.

The input for the plugin consists of an OWL file, a rule file and a mappings file. All three components can be edited in Protégé, using the built-in interface for the ontology and the custom “NoHR Rules” and “NoHR Mappings” tabs, provided by the plugin, for the rule and mapping components. The former (as well as the query panel) comes with a dedicated parser to support the creation of correctly formed rules and queries. The latter allows the creation of mappings based on the user’s specification of what columns from which tables of which database should be combined, where the underlying SQL queries are dynamically generated, based on the structure of the schema, which allows the automatic

[8] The source code can be obtained at https://github.com/NoHRReasoner/NoHR.

[9] http://protege.stanford.edu

[10] http://xsb.sourceforge.net

[11] Similar concepts have been used before for adding database support to rule systems, such as $DLV^{DB}$ [78], and in ontology based data access, such as in ontop [19].

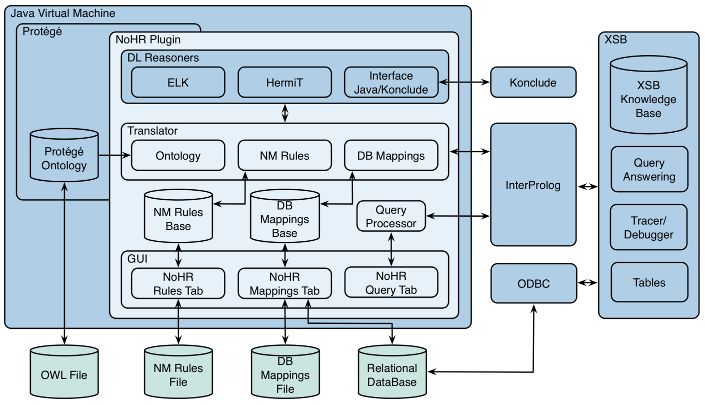


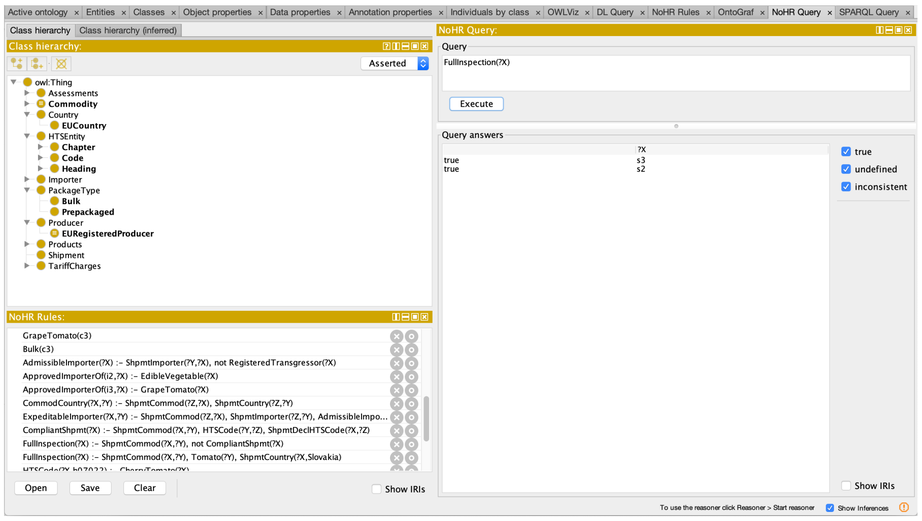
**Fig. 5.** System architecture of NoHR v4.0 with native database support

application of several optimizations to the generated queries. Alternatively, arbitrary SQL queries can be written to take advantage of the capabilities of the specific DBMS at hand for the sake of, e.g., benefiting from using advanced joins and the associated performance gains when querying.

After the inputs (which can be empty) and the first query are provided, the ontology is translated into a set of rules, using one of the provided reasoners, ELK [47], HermiT [34] or Konclude [75], depending on the DL in which the ontology is written. The resulting set is then combined with the rules and mappings provided by the input. This joined result serves as input to XSB Prolog via InterProlog,[12] which is an open-source Java front-end, allowing the communication between Java and a Prolog engine, and the query is sent via the same interface to XSB to be executed. During the execution, mappings are providing facts from the external databases as they are requested in the reasoning process. This procedure is supported by the installed ODBC connections and handled within XSB, thus providing full control over the database access during querying and taking advantage of the built-in optimization to access only the relevant part of the database. Answers are returned to the query processor, which displays them to the user in a table (in the Query Tab). Figure 6 provides an example for the query `FullInspection(?X)`, where we note that variables are denoted with a leading "?" to facilitate distinguishing them from constants, similar to SPARQL. The user may pose further queries, and the system will simply send them directly to XSB, without any repeated preprocessing. If the knowledge base is edited, the system recompiles only the component that was changed.

Different versions of NoHR have been evaluated focussing on different aspects [43, 21, 55, 45], and the main observations are summarized in the following.

[12] http://interprolog.com/java-bridge/



```
owl:Thing
  Assessments
  Commodity
  Country
    EUCountry
  HTSEntity
    Chapter
    Code
    Heading
  Importer
  PackageType
    Bulk
    Prepackaged
  Producer
    EURegisteredProducer
  Products
  Shipment
  TariffCharges
```

```
GrapeTomato(c3)
Bulk(c3)
AdmissibleImporter(?X) :- ShpmtImporter(?Y,?X), not RegisteredTransgressor(?X)
ApprovedImporterOf(i2,?X) :- EdibleVegetable(?X)
ApprovedImporterOf(i3,?X) :- GrapeTomato(?X)
CommodCountry(?X,?Y) :- ShpmtCommod(?Z,?X), ShpmtCountry(?Z,?Y)
ExpeditableImporter(?X,?Y) :- ShpmtCommod(?Z,?X), ShpmtImporter(?Z,?Y), AdmissibleImpo...
CompliantShpmt(?X) :- ShpmtCommod(?X,?Y), HTSCode(?Y,?Z), ShpmtDeclHTSCode(?X,?Z)
FullInspection(?X) :- ShpmtCommod(?X,?Y), not CompliantShpmt(?X)
FullInspection(?X) :- ShpmtCommod(?X,?Y), Tomato(?Y), ShpmtCountry(?X,Slovakia)
```

Query: FullInspection(?X)

|  | ?X |
|---|---|
| true | s3 |
| true | s2 |

**Fig. 6.** Cargo assessment example in the NoHR Protégé plugin

Different ontologies can be preprocessed for querying in a short amount of time (around one minute for SNOMED CT with over 300,000 concepts), and increasing the number of rules only raises the time for translation linearly. As the preprocessing commonly only needs to be done once before queryin, this is less important for the overall performance. In terms of performance of the different used reasoners for preprocessing, it has been shown [55] that ELK is indeed always fastest, so whenever the ontology fits $\mathcal{EL}_{\perp}^{+}$, the dedicated translation module should be used. In between the to general purpose reasoners, HermiT is faster than Konclude on all instances where it does not time out when classifying the given ontology. Konclude then is preferred in the cases where HermiT fails to classify an ontology that does not fit the OWL 2 EL profile (in which ELK cannot be used either).

In comparison to preprocessing times, querying time is in general neglectable. It has been shown that NoHR scales reasonably well for query answering without non-monotonic rules (only slowing down for memory-intensive cases), even for over a million facts/assertions in the ABox, despite being slightly slower on average for OWL QL in comparison to to the other cases, as part of the OWL inferences is encoded in the rule translations directly, and adding rules scales linearly for pre-processing and querying, even for an ontology with many negative inclusions (such as $DL\text{-}Lite_R$).

In addition, with respect to the database component, it has been shown that, if the data is stored in a database and accessed directly during querying instead of being loaded into memory in the form of facts or ontology assertions, preprocessing time and memory consumption substantially reduces, in particular for tuples of higher arity. In terms of querying, on average querying becomes slightly slower, as the connection via ODBC adds an overhead to the query process. However, if advanced mappings are used, which allow outsourcing certain joins over data from XSB to the DBMS, then improvements of considerable margin can be achieved, in particular when advanced database joins reduce the amount of data that needs to be sent to XSB for reasoning.

## 7 Conclusions

In this course, we have provided an overview on the integration of Description Logic ontologies and nonmonotonic rules. For that purpose, we have first recalled the two formalisms and reviewed in detail their characteristics and inherent differences. We have then discussed main criteria based on which such an integration can be achieved and argued that often the right choice depends on the application at hand. To illustrate these ideas, we have presented four concrete approaches and compared them with the help of these established criteria. We note that many existing approaches were left out from our presentation. We refer the interested reader to the references mentioned so far (e.g., [27, 62] provide detailed discussions of related work), as well as the material from previous Reasoning Web lectures with a different focus [25, 50]. We have complemented our considerations on different approaches for such an integration with a more detailed description of one of the reasoning tools, NoHR, that comes with support for databases, robustness to inconsistencies, and fast interactive response times (after a brief preprocessing period), even for larger knowledge bases.

There exists a lot of related work that rather than combining both formalisms aims at combining open and closed world reasoning, by extending one of the two base formalisms with some features from the other. Namely, there is a lot of work on enriching DLs with nonmonotonic reasoning. Description Logics have been extended with default logic [7], with modal operators [23, 49] similar to those used in the rules of Hybrid MKNF KBs, circumscription [13, 51], as well as defeasible logics [16], and rational closure [33]. On the other hand, rules have been extended with existentials in the head, resulting in Datalog$^{+-}$ [17]. While such rule are undecidable in general, a plethora of different restricted such rule fragments has been defined (see, e.g., [8]) allowing to cover a considerable part of the OWL 2 profiles, and for which answer sets semantics [58] and well-founded semantics [35] have been defined.

For future work, considering dynamics in such combinations of DL ontologies and nonmonotonic rules building on previous work [72–74], in particular in the presence of streams [11] and possibly incorporating heterogeneous knowledge [22, 15] seems promising given the huge amounts of data and knowledge that are being created with ever increasing speed and in a variety of formats, for which knowledge-intensive applications are desirable that take advantage of all that information. This certainly is an ambitious objective, but interesting nonetheless.

**Acknowledgements** The author thanks Ricardo Gonçalves and the anonymous reviewers for helpful feedback and acknowledges partial support by FCT projects RIVER (PTDC/CCI-COM/30952/2017) and NOVA LINCS (UIDB/04516/2020).

## References

1. Alberti, M., Knorr, M., Gomes, A.S., Leite, J., Gonçalves, R., Slota, M.: Normative systems require hybrid knowledge bases. In: van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.) Procs. of AAMAS. pp. 1425–1426. IFAAMAS (2012)
2. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. **14**(2), 1–43 (2013)
3. Alviano, M., Calimeri, F., Dodaro, C., Fuscà, D., Leone, N., Perri, S., Ricca, F., Veltri, P., Zangari, J.: The ASP system DLV2. In: LPNMR. LNCS, vol. 10377, pp. 215–221. Springer (2017)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. Artif. Intell. Res. (JAIR) **36**, 1–69 (2009)
5. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 3rd edn. (2010)
6. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Procs. of IJCAI. pp. 364–369. Professional Book Center (2005)
7. Baader, F., Hollunder, B.: Embedding defaults into terminological representation systems. Journal of Automated Reasoning **14**, 149–180 (1995)
8. Baget, J., Leclère, M., Mugnier, M., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. **175**(9-10), 1620–1654 (2011)
9. Bajraktari, L., Ortiz, M., Simkus, M.: Combining rules and ontologies into clopen knowledge bases. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Procs. of AAAI. pp. 1728–1735. AAAI Press (2018)
10. Baral, C., Gelfond, M.: Logic programming and knowledge representation. J. Log. Program. **19/20**, 73–148 (1994)
11. Beck, H., Dao-Tran, M., Eiter, T.: LARS: A logic-based framework for analytic reasoning over streams. Artif. Intell. **261**, 16–70 (2018)
12. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American pp. 96–101 (May 2001)
13. Bonatti, P.A., Lutz, C., Wolter, F.: The complexity of circumscription in DLs. Journal of Artificial Intelligence Research (JAIR) **35**, 717–773 (2009)
14. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier (2004)
15. Brewka, G., Ellmauthaler, S., Gonçalves, R., Knorr, M., Leite, J., Pührer, J.: Reactive multi-context systems: Heterogeneous reasoning in dynamic environments. Artif. Intell. **256**, 68–104 (2018)
16. Britz, K., Casini, G., Meyer, T., Moodley, K., Sattler, U., Varzinczak, I.: Principles of klm-style defeasible description logics. ACM Trans. Comput. Log. **22**(1), 1:1–1:46 (2021)
17. Calì, A., Gottlob, G., Lukasiewicz, T., Pieris, A.: Datalog+/-: A family of languages for ontology querying. In: Datalog. LNCS, vol. 6702, pp. 351–368. Springer (2010)
18. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: Asp-core-2 input language format. Theory Pract. Log. Program. **20**(2), 294–309 (2020)
19. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web **8**(3), 471–487 (2017)

20. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM **43**(1), 20–74 (1996)
21. Costa, N., Knorr, M., Leite, J.: Next step for NoHR: OWL 2 QL. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) Procs. of ISWC. LNCS, vol. 9366, pp. 569–586 (2015)
22. Dao-Tran, M., Eiter, T.: Streaming multi-context systems. In: IJCAI. pp. 1000–1007. ijcai.org (2017)
23. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic **3**(2), 177–225 (2002)
24. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., Schüller, P.: A model building framework for answer set programming with external computations. TPLP **16**(4), 418–464 (2016)
25. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the Semantic Web. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web: 4th International Summer School 2008, Venice, Italy, September 7-11, 2008, Tutorial Lectures. pp. 1–53. Springer (2008)
26. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the Semantic Web. ACM Transactions on Computational Logic **12**, 11:1–11:41 (January 2011)
27. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artificial Intelligence **172**(12–13), 1495–1539 (August 2008)
28. Eiter, T., Simkus, M.: Linking open-world knowledge bases using nonmonotonic rules. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) Procs. of LPNMR. LNCS, vol. 9345, pp. 294–308. Springer (2015)
29. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The potsdam answer set solving collection. AI Commun. **24**(2), 107–124 (2011)
30. van Gelder, A.: The alternating fixpoint of logic programs with negation. In: Principles of Database Systems. pp. 1–10. ACM Press (1989)
31. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM **38**(3), 620–650 (1991)
32. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**, 365–385 (1991)
33. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Semantic characterization of rational closure: From propositional logic to description logics. Artif. Intell. **226**, 1–33 (2015)
34. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
35. Gottlob, G., Hernich, A., Kupke, C., Lukasiewicz, T.: Equality-friendly well-founded semantics and applications to description logics. In: AAAI. AAAI Press (2012)
36. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Hencsey, G., White, B., Chen, Y.R., Kovács, L., Lawrence, S. (eds.) Procs. of WWW. pp. 48–57. ACM (2003)
37. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The RacerPro knowledge representation and reasoning system. Semantic Web journal (2011), to appear. Available at http://www.semantic-web-journal.net/issues

38. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Working Group Note 21 March 2013 (2013), available at https://www.w3.org/TR/sparql11-query/
39. Hitzler, P.: A review of the semantic web field. Commun. ACM **64**(2), 76–83 (2021)
40. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer (Second Edition). W3C (2012)
41. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
42. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Procs. of KR. pp. 57–67. AAAI Press (2006)
43. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) Procs. of ISWC. LNCS, vol. 8218, pp. 216–231 (2013)
44. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Yang, Q., Wooldridge, M.J. (eds.) Procs. of IJCAI. pp. 3098–3105. AAAI Press (2015)
45. Kasalica, V., Gerochristos, I., Alferes, J.J., Gomes, A.S., Knorr, M., Leite, J.: Telco network inventory validation with nohr. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) Procs. of LPNMR. LNCS, vol. 11481, pp. 18–31. Springer (2019)
46. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka, G., Lang, J. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008. AAAI Press (2008)
47. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. Journal of Automated Reasoning **53**, 1–61 (2013)
48. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. **175**(9–10), 1528–1554 (2011)
49. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the semantic web. In: Raedt, L.D., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) Procs. of ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 474–479. IOS Press (2012)
50. Krisnadhi, A.A., Maier, F., Hitzler, P.: OWL and rules. In: Reasoning Web 2011, Springer Lecture Notes in Computer Science (2011), http://knoesis.wright.edu/faculty/pascal/resources/publications/OWL-Rules-2011.pdf, to appear
51. Krisnadhi, A.A., Sengupta, K., Hitzler, P.: Local closed world semantics: Keep it simple, stupid! Tech. rep., Wright State University (2011), available from http://pascal-hitzler.de/resources/publications/GC-DLs.pdf
52. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
53. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7**, 499–562 (July 2006)
54. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) Procs. of IJCAI. Morgan Kaufmann (1991)

55. Lopes, C., Knorr, M., Leite, J.: Nohr: Integrating XSB prolog with the OWL 2 profiles and beyond. In: Procs. of LPNMR. LNCS, vol. 10377, pp. 236–249. Springer (2017)
56. Lukumbuzya, S., Ortiz, M., Simkus, M.: Resilient logic programs: Answer set programs challenged by ontologies. In: AAAI. pp. 2917–2924. AAAI Press (2020)
57. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008). pp. 179–193. No. 5195 in LNAI, Springer (2008)
58. Magka, D., Krötzsch, M., Horrocks, I.: Computing stable models for nonmonotonic existential rules. In: IJCAI. pp. 1031–1038. IJCAI/AAAI (2013)
59. Minker, J., Seipel, D.: Disjunctive logic programming: A survey and assessment. In: Essays in Honour of Robert A. Kowalski, Part I, LNAI 2407. Springer (2002)
60. Morgenstern, L., Welty, C., Boley, H., Hallmark, G. (eds.): RIF Primer (Second Edition). W3C Working Group Note 5 February 2013 (2013), available at https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
61. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles (Second Edition). W3C (2012)
62. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM **57**(5), 93–154 (2010)
63. Motik, B., Sattler, U., Studer, R.: Query-answering for OWL-DL with rules. Journal of Web Semantics **3**(1), 41–60 (2005)
64. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: Rdfox: A highly-scalable RDF store. In: International Semantic Web Conference (2). LNCS, vol. 9367, pp. 3–20. Springer (2015)
65. Patel, C., et al.: Matching patient records to clinical trials using ontologies. In: Aberer, K., Choi, K., Noy, N.F., Allemang, D., Lee, K., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) Procs. of ISWC. LNCS, vol. 4825, pp. 816–829 (2007)
66. Redl, C.: The dlvhex system for knowledge representation: recent advances (system description). Theory Pract. Log. Program. **16**(5-6), 866–883 (2016)
67. Rosati, R.: On the decidability and complexity of integrating ontologies and rules. Journal of Web Semantics **3**(1), 41–60 (2005)
68. Rosati, R.: DL+Log: A tight integration of description logics and disjunctive datalog. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Tenth International Conference on the Principles of Knowledge Representation and Reasoning, KR'06. pp. 68–78. AAAI Press (2006)
69. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach (4th Edition). Pearson (2020)
70. Schreiber, G., Raimond, Y. (eds.): RDF 1.1 Primer. W3C Working Group Note 24 June 2014 (2014), available at https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
71. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics **5**, 51–53 (2007)
72. Slota, M., Leite, J.: Towards closed world reasoning in dynamic open worlds. TPLP **10**(4-6), 547–563 (2010)
73. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP **11**(4-5), 801–819 (2011)
74. Slota, M., Leite, J., Swift, T.: On updates of hybrid knowledge bases composed of ontologies and rules. Artif. Intell. **229**, 33–104 (2015)

75. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. **27**, 78–85 (2014)
76. Sterling, L., Shapiro, E.: The Art of Prolog - Advanced Programming Techniques, 2nd Ed. MIT Press (1994)
77. Swift, T., Warren, D.S.: XSB: extending prolog with tabled logic programming. Theory Pract. Log. Program. **12**(1-2), 157–187 (2012)
78. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. TPLP **8**(2), 129–165 (2008)
79. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) In Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006). pp. 292–297. Springer (2006)
80. Wang, Y., You, J.H., Yuan, L.Y., Shen, Y.D.: Loop formulas for description logic programs. Theory and Practice of Logic Programming **10**(4–6), 531–545 (2010)
81. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: IJCAI. pp. 5511–5519. ijcai.org (2018)